

Basics of PowerShell Looping: Foreach

Rate this article ★★★★★



The Scripting Guys (<https://social.technet.microsoft.com/profile/The+Scripting+Guys>) April 28, 2014

11 (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comments>)

Summary: Microsoft Scripting Guy, Ed Wilson, talks about using the Windows PowerShell **Foreach** statement to loop through a collection.

Share 7

0

0

Microsoft Scripting Guy, Ed Wilson, is here. When the Scripting Wife and I were in Amsterdam, Windows PowerShell MVP, Jeff Wouters, told me that a lot of people he ran across had problems looping through collections with Windows PowerShell. Here is a picture of Jeff and me.



(<https://msdnshared.blob.core.windows.net/media/TNBlogFS/prod.evul.blogs.technet.com/CommunityServer.Blogs.Components.Web>)

Basics of looping

Looping is a fundamental Windows PowerShell concept. Actually, I take that back. It is a fundamental concept of any programming language, even batch languages. So what is the problem?

Most people coming to Windows PowerShell for the first time understand about variables. For example, if I store a value in a variable, and if I want to get at that value, it is no problem. I address the variable as shown here:

```
PS C:\> $a = 5
PS C:\> $b = 6
PS C:\> $c = 7
PS C:\> $a
5
PS C:\> $b
6
PS C:\> $c
7
```

One thing that makes Windows PowerShell easy to use, is that it automatically unravels arrays. An array is when I add more than one thing to a variable. For example, earlier I assigned three values to three variables. Now, I want to add those three variables to a single variable. So I will use **\$d** to hold an array comprised of **\$a**, **\$b**, and **\$c**. In Windows PowerShell, it is easy to see the values. I just call the variable as shown here:

```
PS C:\> $d = $a,$b,$c
PS C:\> $d
5
6
7
```

I can also access the values of the variables by position in the array. The first position is [0], and the last position in our array is [2]. So I can access specific elements from the array by using the position numbers. This is shown here:

```
PS C:\> $d[0]
5
PS C:\> $d[1]
6
PS C:\> $d[2]
7
PS C:\>
```

Walking through the array

Suppose I want to add the number five to each of the three values I have in **\$a**, **\$b**, and **\$c**. If I work with them individually, it is easy. I just do the following:

```
PS C:\> $a + 5
10
PS C:\> $b + 5
11
PS C:\> $c + 5
12
PS C:\>
```

The problem comes with the values I have in my array that is in the **\$d** variable. In Windows PowerShell, if I add 5 to my **\$d** variable, I end up actually adding the value as another element in the array. This is shown here:

```
PS C:\> $d + 5
5
6
7
5
```

To add the number five to each of the elements in the array, I need to walk through the array by using the **Foreach** command. To use the **foreach** command, I need to do three things:

1. I call the **Foreach** command.
2. I use a pair of parentheses, I use a variable for my place holder (enumerator), and I use the variable that is holding the collection.
3. I use a pair of curly braces (script block) that includes the script that does what I want to do.

The placeholder variable I use represents the current item from the collection that I will be working with. The variable only gets a value inside the script block, and it will always be a different item each time I loop through the collection. The **Foreach** command is shown here:

```
Foreach (placeholder variable IN collection)
{
    What I want to do to the placeholder variable
}
```

In my example, the **Foreach** command is shown here:

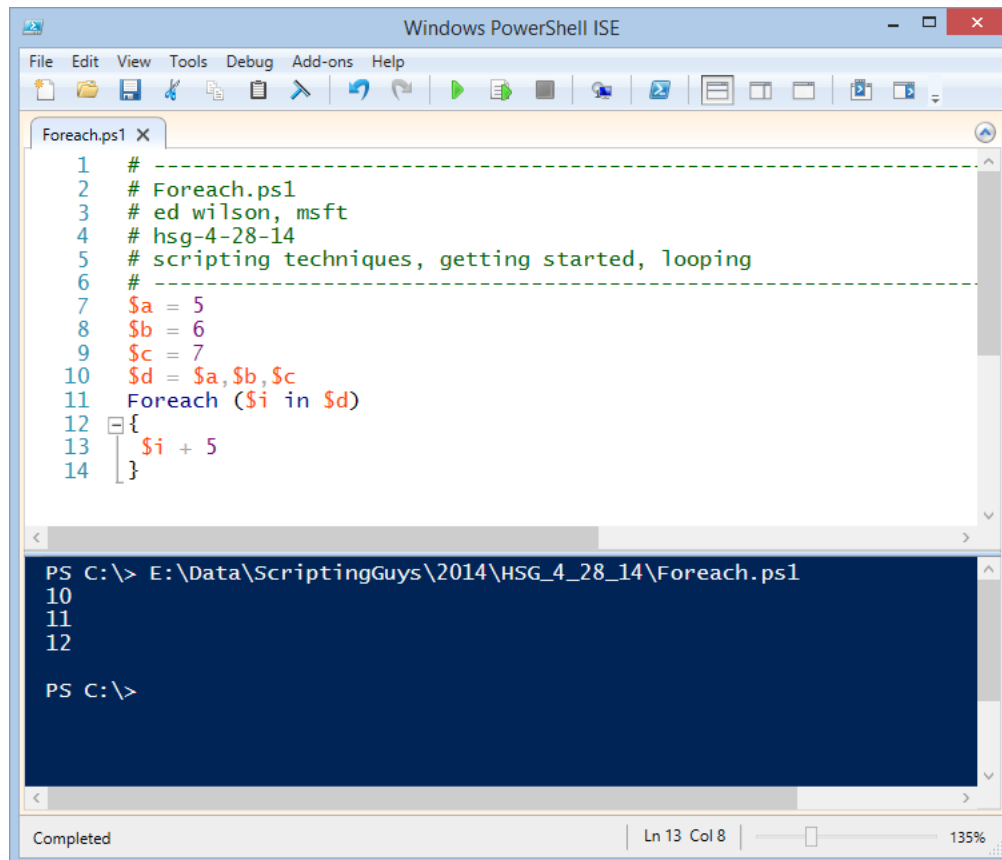
```
Foreach ($i in $d)
{
    $i + 5
```

```
}
```

Here's the entire script:

```
$a = 5  
$b = 6  
$c = 7  
$d = $a,$b,$c  
Foreach ($i in $d)  
{  
    $i + 5  
}
```

The command and the output from the command are shown in the following image:



The screenshot shows the Windows PowerShell ISE interface. The script file 'Foreach.ps1' is open, displaying the following code:

```
1 # -----  
2 # Foreach.ps1  
3 # ed wilson, msft  
4 # hsg-4-28-14  
5 # scripting techniques, getting started, looping  
6 # -----  
7 $a = 5  
8 $b = 6  
9 $c = 7  
10 $d = $a,$b,$c  
11 Foreach ($i in $d)  
12 {  
13     $i + 5  
14 }
```

The console window below the script shows the command being executed and the output:

```
PS C:\> E:\Data\ScriptingGuys\2014\HSG_4_28_14\Foreach.ps1  
10  
11  
12  
PS C:\>
```

The status bar at the bottom indicates 'Completed', 'Ln 13 Col 8', and '135%' zoom.

(<https://msdnshared.blob.core.windows.net/media/TNBlogFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.Web/4-28-14-02.png>)

That is all there is to using **Foreach** to loop through a collection. Looping Week will continue tomorrow when I will talk about using **Foreach-Object** in the pipeline.

I invite you to follow me on Twitter (<http://bit.ly/scriptingguytwitter>) and Facebook (<http://bit.ly/scriptingguyfacebook>). If you have any questions, send email to me at scripter@microsoft.com (<mailto:scripter@microsoft.com>), or post your questions on the Official Scripting Guys Forum (<http://bit.ly/scriptingforum>). See you tomorrow. Until then, peace.

Ed Wilson, Microsoft Scripting Guy

Tags [getting started](https://blogs.technet.microsoft.com/heyscriptingguy/tag/getting-started/) (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/getting-started/>) [looping](https://blogs.technet.microsoft.com/heyscriptingguy/tag/looping/) (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/looping/>) [Scripting Guy!](https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-guy/) (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-guy/>) [scripting techniques](https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-techniques/) (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-techniques/>) [Windows PowerShell](https://blogs.technet.microsoft.com/heyscriptingguy/tag/windows-powershell/) (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/windows-powershell/>)

Comments (11)

Name * Email * Website ☐ Save my name, email, and website in this browser for the next time I comment.[Post Comment](#)*Vidyasagar Machupalli* (<https://social.technet.microsoft.com/profile/Vidyasagar+Machupalli>)October 15, 2018 at 8:40 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121493>)

Good ps1 script about looping...Thanks

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121493#respond>)*Anonymous*October 15, 2018 at 8:40 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121503>)

In this example '\$i' is just a placeholder.

It's just a variable named 'i'.

You could call it : "\$AnythingYouLike"

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121503#respond>)*Anonymous*October 15, 2018 at 8:40 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121513>)

Now that's a great starters example for everyone who wants to learn looping in PowerShell.

Maybe you could explain to everyone (inc. Jeff 🙄) what the difference is between foreach and foreach-object.

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121513#respond>)*Victor Ashiedu* (<https://social.technet.microsoft.com/profile/Victor+Ashiedu>)October 15, 2018 at 8:40 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121523>)

ForEach is very useful but I'll like to know the difference with ForEach-Object

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121523#respond>)*Joe*April 28, 2014 at 4:14 pm (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121543>)

Does it need to say Foreach (\$i in \$d)? Where does the "\$i" come from? I understand in C and C++ you have to declare it in a loop, (i=0, i>10,i++). I am not sure where and how the \$i works in powershell yet.

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121543#respond>)*Joe*April 29, 2014 at 12:09 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121473>)

Thanks Gaff, so you never have to declare it? you can just use whatever you want as long as you reference the name of the array?

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121473#respond>)*markus*May 10, 2014 at 8:23 am (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121553>)

Do I need to declare a variable first to be able to use a Foreach-Object cmdlet?

Can I use for example: get-moverequest | select identity | % {get-moverequeststatistics}

In that example, the mandatory Parameter "identity" isn't parsed to the get-moverequeststatistics... But why?

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121553#respond>)*zalek*November 13, 2014 at 11:21 pm (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121533>)

why the output file was not changed? Here is my code:

```
$batch = Get-Content $file
foreach ($i in $batch)
{
    if ($i.StartsWith('02 '))
    {
        $i = '05 ' + $i.Substring(3)}
    }
}
```

Set-Content \$outFile \$batch

The output file was the same as the input file, but the code changes '02 ' to '05 '.

Thanks,

Zalek

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121533#respond>)

*JoeT*

January 28, 2015 at 11:11 pm (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-121483>)
 I'm totally new to PS. I'm trying to reset all 18K user account password to a uniquely generated password for each account that I have populated in a .csv file.

Here's the script that I'm using but I got the following error message – "Unexpected token 'in' in expression or statement". What did I do wrong? Thanks in advance for helping!

```
connect-msolservice
```

```
$Users = Import-Csv C:\Users\xxx\Documents\passwordTest.csv
foreach($User in $Users){
set-msoluserpassword -userprincipalname $User.UserPrincipalName -newpassword $User.NewPassword
}
```

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=121483#respond>)

*vishnu*

July 27, 2015 at 1:27 pm (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-150001>)
 Greatttt . I am just a beginner. i was struggling to get the concept of this. This helped me lot.

Thanks,

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=150001#respond>)

*James III*

January 5, 2016 at 6:29 pm (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/#comment-167261>)
 Isn't easier to write the script as follows:

```
$digits = 2,3,4,5
foreach($digit in $digits) {
$digit + 3
}
```

Reply (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/04/28/basics-of-powershell-looping-foreach/?replytocom=167261#respond>)

Follow Us



(<https://blogs.technet.microsoft.com/heyscriptingguy/feed/>)

Search MSDN with Bing

☐ Search this blog ☒ Search all blogs

Top Server & Tools Blogs

ScottGu's Blog (<http://weblogs.asp.net/scottgu>)

Brad Anderson's "In the Cloud" Blog (https://blogs.technet.microsoft.com/in_the_cloud/)

Brian Harry's Blog (<http://blogs.msdn.microsoft.com/bharry/>)

Steve "Guggs" Guggenheimer's Blog (<https://blogs.msdn.microsoft.com/stevengu/>)



Subscribe
 (<https://blogs.technet.microsoft.com/heyscriptingguy/feed/>)

Recent Posts

- PowerTip: Use PowerShell to pick a random name from a list (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/09/15/powertip-use-powershell-to-pick-a-random-name-from-a-list/>) September 15, 2018
- Using PowerShell to create a folder of Demo data (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/09/15/using-powershell-to-create-a-folder-of-demo-data/>) September 15, 2018
- PowerTip: Turn off the power to your computer with PowerShell (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/07/16/powertip-turn-off-the-power-to-your-computer-with-powershell/>) July 16, 2018
- Parse HTML and pass to Cognitive Services Text-to-Speech (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/07/16/parse-html-and-pass-to-cognitive-services-text-to-speech/>) July 16, 2018

Tags

Scripting Guy! (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-guy/>)

Windows PowerShell (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/windows-powershell/>)

scripting techniques (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/scripting-techniques/>)

PowerTip (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/powertip/>)

guest blogger (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/guest-blogger/>)

VBScript (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/vbscript/>)

getting started (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/getting-started/>)

Weekend Scripter (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/weekend-scripter/>)

Sean Kearney (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/sean-kearney/>)

Office (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/office/>)

Active Directory (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/active-directory/>)

operating system (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/operating-system/>)

storage (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/storage/>) WMI (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/wmi/>)

files (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/files/>) text files (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/text-files/>)

community (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/community/>)

desktop management (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/desktop-management/>)

2011 Scripting Games (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/2011-scripting-games/>)

2012 Scripting Games (<https://blogs.technet.microsoft.com/heyscriptingguy/tag/2012-scripting-games/>)

Related Resources

Script Center Home (<http://technet.microsoft.com/en-us/scriptcenter/default.aspx>)

Scripting Library (<http://technet.microsoft.com/en-us/library/cc498722.aspx>)

Learn to Script (<http://technet.microsoft.com/en-us/scriptcenter/dd793612.aspx>)

Script Repository (<http://www.microsoft.com/technet/scriptcenter/scripts/default.aspx?mfr=true>)

Scripting Forum (<http://social.technet.microsoft.com/Forums/en-US/ITCG/threads>)

2012 Scripting Games (<http://blogs.technet.com/b/heyscriptingguy/archive/2012/02/04/the-2012-windows-powershell-scripting-games-all-links-on-one-page.aspx>)

Archives

September 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/09/>) (2)

July 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/07/>) (2)

June 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/06/>) (4)

May 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/05/>) (5)

March 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/03/>) (3)

February 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/02/>) (4)

December 2017 (<https://blogs.technet.microsoft.com/heyscriptingguy/2017/12/>) (7)

October 2017 (<https://blogs.technet.microsoft.com/heyscriptingguy/2017/10/>) (2)

August 2017 (<https://blogs.technet.microsoft.com/heyscriptingguy/2017/08/>) (2)

July 2017 (<https://blogs.technet.microsoft.com/heyscriptingguy/2017/07/>) (1)

All of 2018 (<https://blogs.technet.microsoft.com/heyscriptingguy/2018/>) (20)

All of 2017 (<https://blogs.technet.microsoft.com/heyscriptingguy/2017/>) (30)

All of 2016 (<https://blogs.technet.microsoft.com/heyscriptingguy/2016/>) (244)

All of 2015 (<https://blogs.technet.microsoft.com/heyscriptingguy/2015/>) (726)

All of 2014 (<https://blogs.technet.microsoft.com/heyscriptingguy/2014/>) (740)

All of 2013 (<https://blogs.technet.microsoft.com/heyscriptingguy/2013/>) (757)

All of 2012 (<https://blogs.technet.microsoft.com/heyscriptingguy/2012/>) (624)

All of 2011 (<https://blogs.technet.microsoft.com/heyscriptingguy/2011/>) (442)

All of 2010 (<https://blogs.technet.microsoft.com/heyscriptingguy/2010/>) (432)

All of 2009 (<https://blogs.technet.microsoft.com/heyscriptingguy/2009/>) (285)

All of 2008 (<https://blogs.technet.microsoft.com/heyscriptingguy/2008/>) (181)

All of 2007 (<https://blogs.technet.microsoft.com/heyscriptingguy/2007/>) (243)

All of 2006 (<https://blogs.technet.microsoft.com/heyscriptingguy/2006/>) (244)

All of 2005 (<https://blogs.technet.microsoft.com/heyscriptingguy/2005/>) (248)

All of 2004 (<https://blogs.technet.microsoft.com/heyscriptingguy/2004/>) (100)

Privacy (<https://privacy.microsoft.com>) Terms of Use (<https://msdn.microsoft.com/cc300389>)

Third-Party Links (<https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/EN-US.aspx>)

© 2018 Microsoft